Towards μs Tail Latency and Terabit Ethernet: Disaggregating the Host Network Stack

Qizhe Cai
Midhul Vuppalapati
Jaehyun Hwang
Christos Kozyrakis
Rachit Agarwal
Limitations of Existing Host Network Stacks

Inefficient Packet Processing Pipeline

Diagram:
- User space: Apps
- Kernel space: Linux Network Stack
- H/W: NIC

Icon made by Freepik from www.flaticon.com
Limitations of Existing Host Network Stacks

Frequently Cited Problems

Inefficient Packet Processing Pipeline
Limitations of Existing Host Network Stacks

Frequently Cited Problems

- Inefficient Packet Processing Pipeline
- Poor Performance Isolation
Limitations of Existing Host Network Stacks

- Inefficient Packet Processing Pipeline
- Poor Performance Isolation
- Rigid & Complex Implementation
Limitations of Existing Host Network Stacks

Frequently Cited Problems

- Inefficient Packet Processing Pipeline
- Poor Performance Isolation
- Rigid & Complex Implementation
- Inefficient Transport Protocols

Limitations of Existing Host Network Stacks

Frequently Cited Problems
- Inefficient Packet Processing Pipeline
- Poor Performance Isolation
- Rigid & Complex Implementation
- Inefficient Transport Protocols

Many interesting debates on various design aspects
Debates on various design aspects
Debates on various design aspects
Debates on various design aspects

- Interface
  - Streaming
  - RPC

- Semantics
  - Synchronous
  - Asynchronous
Debates on various design aspects

**Interface**
- Streaming
- RPC

**Semantics**
- Synchronous
- Asynchronous

**Placement**
- In-kernel
- Userspace
- Hardware
This paper: many limitations of the Host Network Stack are *not* rooted in Interface, Semantics or Placement but rather in its Core Architecture.
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a “pipe” abstraction.
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a "pipe" abstraction
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a “pipe” abstraction
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a "pipe" abstraction

Dedicated
Each application/socket has an independent instance of packet processing pipeline
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a “pipe” abstraction

- **Dedicated**
  Each application/socket has an independent instance of packet processing pipeline

- **Tightly Integrated**
  Packet processing coupled to application cores
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a “pipe” abstraction

- **Dedicated**
  Each application/socket has an independent instance of packet processing pipeline

- **Tightly Integrated**
  Packet processing coupled to application cores

- **Static**
  Host resource provisioning determined at pipe creation (Independent of other pipes and resource availability)
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a “pipe” abstraction

- **Dedicated**
  - Each application/socket has an independent instance of packet processing pipeline

- **Tightly Integrated**
  - Packet processing coupled to application cores

- **Static**
  - Host resource provisioning determined at pipe creation (Independent of other pipes and resource availability)

Dedicated, Tightly Integrated, and Static Pipelines: preclude network stacks from exploiting capabilities of modern hardware
This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware
This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline
This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline

Prototype NetChannel Implementation in the Linux Network Stack
Demonstrate new operating points through experimental evaluation
This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline

Prototype NetChannel Implementation in the Linux Network Stack
Demonstrate new operating points through experimental evaluation
Problem with Static and Dedicated Pipes

Linux Network Stack

socket queues

User space

Kernel space

App

Interface-level Processing
Protocol Processing
Netdevice Subsystem

H/W

NIC

Network

100Gbps
Problem with Static and Dedicated Pipes

Long Flows (link bandwidth > per-core throughput)
Problem with Static and Dedicated Pipes

Linux Network Stack

- Network
- NIC
- 100Gbps
- Interface-level Processing
- Protocol Processing
- Netdevice Subsystem
- socket queues
- User space
- Kernel space
- H/W

Long Flows (link bandwidth > per-core throughput)

<table>
<thead>
<tr>
<th>Category</th>
<th>Fraction of CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface-level Processing</td>
<td>0.70</td>
</tr>
<tr>
<td>Network Layer Processing</td>
<td>0.53</td>
</tr>
<tr>
<td>Other</td>
<td>0.35</td>
</tr>
</tbody>
</table>

CPU Breakdown

Icon made by Freepik from www.flaticon.com
Problem with Static and Dedicated Pipes

Long Flows (link bandwidth > per-core throughput)

CPU Breakdown

Fraction of CPU

0.00 0.18 0.35 0.53 0.70

Interface-level Processing Network Layer Processing Other

Network

100Gbps NIC

H/W

Interface-level Processing Protocol Processing Netdevice Subsystem

Kernel space User space

socket queues

App

Linux Network Stack

Network Layer Processing

Other
Problem with Static and Dedicated Pipes

Linux Network Stack

- App
- User space
- Kernel space
- H/W
- NIC
- Network
- Interface-level Processing
- Protocol Processing
- Netdevice Subsystem

Long Flows (link bandwidth > per-core throughput)

CPU Breakdown:
- Interface-level Processing
- Network Layer Processing
- Other

Fraction of CPU:
- 0.00
- 0.18
- 0.35
- 0.53
- 0.70

100Gbps

Icon made by Freepik from www.flaticon.com
Problem with Static and Dedicated Pipes

Resources for Interface-level Processing limited by number of application cores

Long Flows (link bandwidth > per-core throughput)
Problem with Static and Dedicated Pipes

Resources for Interface-level Processing limited by number of application cores

Ideal: Dynamically scale Interface-level processing based on resource availability
Problem with Static and Dedicated Pipes

Linux Network Stack

- App
- User space
- Kernel space
- socket queues
- H/W

NIC

- Interface-level Processing
- Protocol Processing
- Netdevice Subsystem

100Gbps

Network
Problem with Static and Dedicated Pipes

Short Messages (link bandwidth > per-core throughput)
Problem with Static and Dedicated Pipes

Linux Network Stack

- App
- socket queues
- User space
- Interface-level Processing
- Protocol Processing
- Netdevice Subsystem
- Kernel space
- H/W
- NIC
- 100Gbps
- Network

Short Messages (link bandwidth > per-core throughput)

Fraction of CPU

- 0.00
- 0.15
- 0.30
- 0.45
- 0.60

- Interface-level Processing
- Network Layer Processing
- Other

CPU Breakdown

Problem with Static and Dedicated Pipes
Problem with Static and Dedicated Pipes

Short Messages (link bandwidth > per-core throughput)

CPU Breakdown

- Interface-level Processing
- Network Layer Processing
- Other

Fraction of CPU

- 0.00
- 0.15
- 0.30
- 0.45
- 0.60

100Gbps

Network

 NIC

H/W

Kernel space

User space

socket queues

Linux Network Stack

App

Protocol Processing

Netdevice Subsystem

Interface-level Processing

Icon made by Freepik from www.flaticon.com
Problem with Static and Dedicated Pipes

Linux Network Stack

- App
- User space
- Kernel space
- H/W
- Interface-level Processing
- Protocol Processing
- Netdevice Subsystem

100Gbps

Short Messages (link bandwidth > per-core throughput)

- Fraction of CPU
- Interface-level Processing
- Network Layer Processing
- Other

CPU Breakdown

Icon made by Freepik from www.flaticon.com

Problem with Static and Dedicated Pipes
Problem with Static and Dedicated Pipes

Linux Network Stack

Short Messages (link bandwidth > per-core throughput)

Resources for Network Layer Processing limited by number of connections
Resources for Network Layer Processing limited by number of connections

Ideal: Dynamically scale Network Layer Processing based on resource availability
Problem with Static and Dedicated Pipes

Long Flows (link bandwidth > per-core throughput)

Short Messages (link bandwidth > per-core throughput)
Problem with Static and Dedicated Pipes

Long Flows (link bandwidth > per-core throughput)

Need more resources for Interface-level Processing

Short Messages (link bandwidth > per-core throughput)
Problem with Static and Dedicated Pipes

Long Flows (link bandwidth > per-core throughput)
Need more resources for Interface-level Processing

Short Messages (link bandwidth > per-core throughput)
Need more resources for Network Layer Processing
Different applications can have bottlenecks at different parts of the pipeline.

- **Long Flows (link bandwidth > per-core throughput)***
  - Need more resources for **Interface-level Processing**

- **Short Messages (link bandwidth > per-core throughput)***
  - Need more resources for **Network Layer Processing**
Problem with Static and Dedicated Pipes

Different applications can have bottlenecks at different parts of the pipeline

Ideal: Network stack should be able to independently allocate resources for different applications at different parts of the pipeline

Long Flows (link bandwidth > per-core throughput)
Need more resources for **Interface-level Processing**

Short Messages (link bandwidth > per-core throughput)
Need more resources for **Network Layer Processing**
Problem with Tightly Integrated Pipes

User space

Kernel space

H/W

NIC

100Gbps

Network

socket queues

Interface-level Processing

Protocol Processing

Netdevice Subsystem

socket queues

Interface-level Processing

Protocol Processing

Netdevice Subsystem

L-app

T-app
Problem with Tightly Integrated Pipes

Tail Latency (µs)

Isolated

Co-located

L-app

T-app

User space

Kernel space

socket queues

Interface-level Processing

Protocol Processing

Netdevice Subsystem

H/W

NIC

100Gbps

Network
Problem with Tightly Integrated Pipes

- L-app
- T-app

User space

- socket queues
- Interface-level Processing
- Protocol Processing
- Netdevice Subsystem

Kernel space

- Interface-level Processing
- Protocol Processing
- Netdevice Subsystem

H/W

- NIC
- 100Gbps

Network

Tail Latency (µs)

- Isolated
- Co-located

Icon made by Freepik from www.flaticon.com
Problem with Tightly Integrated Pipes

Prioritization mechanisms at NetDevice Subsystem & CPU scheduler do not solve the problem.
Problem with Tightly Integrated Pipes

Prioritization mechanisms at NetDevice Subsystem & CPU scheduler do not solve the problem
Problem with Tightly Integrated Pipes

Prioritization mechanisms at NetDevice Subsystem & CPU scheduler do not solve the problem
Problem with Tightly Integrated Pipes

Network Layer Processing coupled to core on which application runs
Problem with Tightly Integrated Pipes

Network Layer Processing coupled to core on which application runs

Ideal: Decouple Network Layer Processing from application cores
Rearchitecture is inevitable for Terabit Ethernet
Rearchitecture is inevitable for Terabit Ethernet

For $≥ 100\text{Gbps}$ networks, recent works have shown that bottlenecks have shifted to the Host

<table>
<thead>
<tr>
<th>SIGCOMM’21</th>
<th>Understanding Host Network Stack Overheads</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGCOMM’20</td>
<td>Swift: Delay is Simple and Effective for Congestion Control in the Datacenter</td>
</tr>
<tr>
<td>SIGCOMM’18</td>
<td>Understanding PCIe performance for end host networking</td>
</tr>
</tbody>
</table>
Rearchitecture is inevitable for Terabit Ethernet

For $≥ 100\text{Gbps}$ networks, recent works have shown that bottlenecks have shifted to the Host

- **SIGCOMM’21**: Understanding Host Network Stack Overheads
- **SIGCOMM’20**: Swift: Delay is Simple and Effective for Congestion Control in the Datacenter
- **SIGCOMM’18**: Understanding PCIe performance for end host networking

**Dedicated, Tightly Integrated Static Pipelines**

**Internet / Early generation Datacenters**

✔️
Rearchitecture is inevitable for Terabit Ethernet

For $\geq 100\text{Gbps}$ networks, recent works have shown that bottlenecks have shifted to the Host

| SIGCOMM’21 | Understanding Host Network Stack Overheads |
| SIGCOMM’20 | Swift: Delay is Simple and Effective for Congestion Control in the Datacenter |
| SIGCOMM’18 | Understanding PCIe performance for end host networking |

<table>
<thead>
<tr>
<th>Dedicated, Tightly Integrated Static Pipelines</th>
<th>Internet / Early generation Datacenters</th>
<th>Today’s Datacenters us-latency, 100s of Gbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Green Checkmark]</td>
<td>![Green Checkmark]</td>
<td>![Red X]</td>
</tr>
</tbody>
</table>

Today’s network stacks are on the brink of breakdown
Rearchitecture is inevitable for Terabit Ethernet

For $\geq 100$Gbps networks, recent works have shown that bottlenecks have shifted to the Host

<table>
<thead>
<tr>
<th>SIGCOMM’21</th>
<th>Understanding Host Network Stack Overheads</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGCOMM’20</td>
<td>Swift: Delay is Simple and Effective for Congestion Control in the Datacenter</td>
</tr>
<tr>
<td>SIGCOMM’18</td>
<td>Understanding PCIe performance for end host networking</td>
</tr>
</tbody>
</table>

Dedicated, Tightly Integrated Static Pipelines

Internet / Early generation Datacenters

Today’s Datacenters
us-latency, 100s of Gbps

Today’s network stacks are on the brink of breakdown

Rearchitecture is inevitable
This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline

Prototype NetChannel Implementation in the Linux Network Stack
Demonstrate new operating points through experimental evaluation
Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers
Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

Linux Network Stack

H/W

NIC

Network
Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

Linux Network Stack

H/W

NIC

Network

socket queues

User space

Kernel space

Dedicated - Shared
Shared resources across pipes

Icon made by Freepik from www.flaticon.com

Icon made by Pixel Perfect from www.flaticon.com
Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

Linux Network Stack

H/W NIC Network

User space

socket queues

Kernel space

Dedicated - Shared
Shared resources across pipes
Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

- **Dedicated** - **Shared**
  - Shared resources across pipes

- **Tightly Integrated** - **Loosely Coupled**
  - Different parts of pipeline are decoupled

Linux Network Stack

- User space
  - socket queues
- Kernel space
- H/W
- NIC
- Network
Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

- **Dedicated** - Shared
  - Shared resources across pipes

- **Tightly Integrated** - Loosely Coupled
  - Different parts of pipeline are decoupled

Diagram:
- Linux Network Stack
  - User space
  - Kernel space
  - Socket queues
  - NIC
  - H/W
  - Network
Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

- **Dedicated** - **Shared**
  - Shared resources across pipes

- **Tightly Integrated** - **Loosely Coupled**
  - Different parts of pipeline are decoupled

- **Static** - **Dynamic**
  - Independent scaling of resources allocated to different parts of the pipeline (Based on resource availability and other pipes)
Disaggregating the Host Network Stack

**Key Idea:** Disaggregate packet processing pipeline into separate layers

- **Dedicated-Shared**
  - Shared resources across pipes

- **Tightly Integrated-Loosely Coupled**
  - Different parts of pipeline are decoupled

- **Static-Dynamic**
  - Independent scaling of resources allocated to different parts of the pipeline (Based on resource availability and other pipes)

**Linux Network Stack**

- **User space**
  - Socket queues

- **Kernel space**

**H/W**

- **NIC**

**Network**
Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

- **Dedicated** - Shared
  - Shared resources across pipes

- **Tightly Integrated** - Loosely Coupled
  - Different parts of pipeline are decoupled

- **Static** - Dynamic
  - Independent scaling of resources allocated to different parts of the pipeline (Based on resource availability and other pipes)
NetChannel Architecture

NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers

- **User space**
  - App
  - App

- **Kernel space**
  - Virtual Network System (VNS)
  - NetScheduler
  - NetDriver

- **H/W**
  - NIC
  - Network
NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers

Virtual Network System (VNS)

NetScheduler

NetDriver

H/W

NIC

Network
NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers

- **Virtual Network System (VNS)**
- Provide **virtual** interfaces (e.g., socket, RPC, ..)

NetChannel Architecture
NetChannel Architecture

NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers

- Virtual Network System (VNS)
  - Provide virtual interfaces (e.g., socket, RPC, ..)
  - Decouples Interface-level Processing

![Diagram of NetChannel Architecture]

- Virtual Network System (VNS)
- NetScheduler
- NetDriver
- H/W
- NIC
- Network

User space

Kernel space

Interface-level Processing threads
NetChannel Architecture

NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers

- **Virtual Network System (VNS)**
  - Provide virtual interfaces (e.g., socket interface)
  - Decouples Interface-level Processing

- **NetDriver**
  - Abstract the network as a multi-queue “device”
    - through channel abstraction
  - Enables easy integration of new transports
  - Decouples Network Layer Processing
NetChannel Architecture

NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers

- **Virtual Network System (VNS)**
  - Provide *virtual* interfaces (e.g., socket interface)
  - Enables decoupling Interface-level Processing

- **NetScheduler**
  - Multiplex/Demultiplex data
    - between virtual interfaces and channels
  - Enables flexible scheduling policies

- **NetDriver**
  - Abstract the network as a multi-queue “device”
    - through *channel* abstraction
  - Enables easy integration of new transports
  - Decouples Network Layer Processing
NetChannel Architecture

NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers

- **Virtual Network System (VNS)**
  - Provide virtual interfaces (e.g., socket interface)
  - Enables decoupling Interface-level Processing

- **NetScheduler**
  - Multiplex/Demultiplex data
    - between virtual interfaces and channels
  - Enables flexible scheduling policies

- **NetDriver**
  - Abstract the network as a multi-queue “device”
    - through channel abstraction
  - Enables easy integration of new transports
  - Decouples Network Layer Processing
NetChannel Architecture

NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers

- **Virtual Network System (VNS)**
  - Provide virtual interfaces (e.g., socket interface)
  - Enables decoupling Interface-level Processing

- **NetScheduler**
  - Multiplex/Demultiplex data
    - between virtual interfaces and channels
  - Enables flexible scheduling policies

- **NetDriver**
  - Abstract the network as a multi-queue “device”
    - through channel abstraction
  - Enables easy integration of new transports
  - Decouples Network Layer Processing
NetChannel End-to-End Operation

(Note: One can define different NetScheduler policies)
NetChannel End-to-End Operation

(Note: One can define different NetScheduler policies)
NetChannel End-to-End Operation

(Note: One can define different NetScheduler policies)
NetChannel End-to-End Operation

(Note: One can define different NetScheduler policies)
NetChannel End-to-End Operation

(Note: One can define different NetScheduler policies)
NetChannel End-to-End Operation

(Note: One can define different NetScheduler policies)
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
Piggybacks on transport-level flow control to apply back pressure
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
Piggybacks on transport-level flow control to apply back pressure
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
Piggybacks on transport-level flow control to apply back pressure
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
Piggybacks on transport-level flow control to apply back pressure
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

Per-virtual socket response queues associated with each channel
Piggybacks on transport-level flow control to apply back pressure
For persistent HoL blocking: Virtual interface level flow control mechanism (see paper for details)
This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline

Prototype NetChannel Implementation in the Linux Network Stack
Demonstrate new operating points through experimental evaluation
Evaluation Setup

- Implemented as a kernel module in Linux
- To push the bottleneck to the Host Network Stack
  - Two 32-core servers directly connected 200Gbps (8 cores per NUMA node)
  - Minimal application-level processing
- Demonstrate NetChannel achieves new operating points
  - Saturate a 200Gbps using a single socket
  - Scale short message processing throughput linearly
  - Achieve μs-scale tail latency for L-apps when colocated with T-apps
- Evaluation on real-world applications
  - Redis: in-memory database
  - SPDK: remote storage stacks
Evaluation Setup

• Implemented as a kernel module in Linux

• To push the bottleneck to the Host Network Stack
  • Two 32-Core servers directly connected 200Gbps (8 cores per NUMA node)
  • Minimal application-level processing

• **Demonstrate NetChannel achieves new operating points**
  • Saturate a 200Gbps using a single socket
  • Scale short message processing throughput linearly
  • Achieve µs-scale tail latency for L-apps when colocated with T-apps

• Evaluation on real-world applications
  • Redis: in-memory database
  • SPDK: remote storage stacks

See Paper for more detailed Evaluation
NetChannel: Towards Terabit Ethernet

NetScheduler

NetDriver

VNS

App

Long flow

NIC

200 Gbps

Network
NetChannel: Towards Terabit Ethernet

![Diagram of NetChannel system](image)

- **App**
- **VNS**
  - **NetScheduler**
  - **NetDriver**
- **NIC**: 200 Gbps
- **Network**

**Long flow**
NetChannel: Towards Terabit Ethernet

VNS
NetScheduler
NetDriver

App

Long flow

NIC
200 Gbps

Network
NetChannel: Towards Terabit Ethernet

NetChannel enables saturating a Terabit Ethernet link with a single socket.
NetChannel: Towards Terabit Ethernet

NetChannel enables saturating a Terabit Ethernet link with a single socket.
NetChannel: Towards Terabit Ethernet

NetChannel enables saturating a Terabit Ethernet link with a single socket.
NetChannel: Towards Terabit Ethernet

NetChannel enables saturating a Terabit Ethernet link with a single socket.
NetChannel: Towards Terabit Ethernet

NetChannel enables saturating a Terabit Ethernet link with a single socket
NetChannel: Towards Terabit Ethernet

NetChannel enables saturating a Terabit Ethernet link with a single socket.

![Diagram of NetChannel](Image)

- **App**
- **VNS**
- **NetScheduler**
- **NetDriver**
- **NIC**
- **Network**

Throughput(Gbps)

<table>
<thead>
<tr>
<th>Linux</th>
<th>Linux + NetChannel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
</tr>
</tbody>
</table>

Throughput(Gbps)

<table>
<thead>
<tr>
<th>#channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

Total throughput: Linux 50 Gbps, Linux + NetChannel 200 Gbps

Throughput-per-core: Linux 50 Gbps, Linux + NetChannel 50 Gbps

NetChannel makes it possible to saturate a Terabit Ethernet link with a single socket.
NetChannel: Towards Terabit Ethernet

NetChannel enables saturating a Terabit Ethernet link with a single socket.
NetChannel: Towards Terabit Ethernet

NetChannel enables saturating a Terabit Ethernet link with a single socket.
NetChannel: Towards Terabit Ethernet

NetChannel enables saturating a Terabit Ethernet link with a single socket

NetChannel enables linear scalability of throughput for short messages
NetChannel: Towards μs Tail Latency
NetChannel: Towards µs Tail Latency
NetChannel: Towards µs Tail Latency
NetChannel: Towards µs Tail Latency

![Diagram showing NetChannel architecture with L-app and T-app connections through NetScheduler and NetDriver, with NIC 200 Gbps and network connections.

The graph on the right shows tail latency comparison between isolated and co-located scenarios:
- Linux: 35 µs
- Linux + NetChannel: 1 µs (17x improvement)

Icon made by Freepik from www.flaticon.com
NetChannel: Towards µs Tail Latency

NetChannel enables µs-scale Tail Latency for L-apps even when co-located with T-apps

Tail Latency (us)

- Isolated
- Co-located

17x improvement

Linux
Linux + NetChannel
Future Directions

NetChannel Architecture

Realizing new NetScheduler policies
Future Directions

NetChannel Architecture

Realizing new NetScheduler policies

Integrating new transport protocols
Future Directions

NetChannel Architecture

- Realizing new NetScheduler policies
- Integrating new transport protocols
- Realizing NetChannel in userspace / hardware

VNS

NetScheduler

NetDriver

App

App

NIC

Network
Future Directions

NetChannel Architecture

Realizing new NetScheduler policies
Integrating new transport protocols
Realizing NetChannel in userspace / hardware

https://github.com/Terabit-Ethernet/NetChannel